68

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
08/820,181	03/14/1997	GEORGE WILLIAM WILHELM JR.	EN995139	2588

7590 01/02/2002

SHELLEY M BECKSTRAND
314 MAIN STREET
OWEGO, NY 13827

EXAMINER	
BANANKHAH, MAJID A	
ART UNIT	**PAPER NUMBER**
2151	

DATE MAILED: 01/02/2002

Please find below and/or attached an Office communication concerning this application or proceeding.

PTO-90C (Rev. 07-01)

**BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES**

Paper No. 11

Application Number: 08/820,181
Filing Date: March 14,1997
Appellant(s): G.W. Wilhelm

Shelly M. Beckstrand
For Appellant

EXAMINER'S ANSWER

MAILED
JAN 02 2002
Technology Center 2100

This is in response to appellant's brief on appeal filed September 28, 2001.

(1) *Real Party in Interest*

A statement identifying the real party in interest is contained in the brief.

(2) *Related Appeals and Interferences*

A statement identifying the related appeals and interferences which will directly affect or be directly affected by or have a bearing on the decision in the pending appeal is contained in the brief.

(3) *Status of Claims*

The statement of the status of the claims contained in the brief is not correct.
The status of all claims are :

1. Claimed canceled: None
2. Claims withdrawn from consideration but not canceled: None
3. Claims pending: 1-8
4. Claims allowed: Claim 8
5. Claims rejected: Claims 1-7

(4) *Status of Amendments*

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) *Summary of Invention*

The summary of invention contained in the brief is correct.

(6) *Issues*

The appellant's statement of the issues in the brief is not correct.

The correct issue is:
Whether claims 1 through 7 are unpatentable under 35 U.S.C. 103 over Davidson et al. (U.S. Patent 5,630,136), in view of Periwal et al. (U.S. Patent 5,644,768).

(7) *Grouping of Claims*

The grouping of claim as stated by appellant is correct.

(8) Claims Appealed

appellant's statement of the grouping of claims in the brief is correct.

(9) Prior Art of Record

The following is a listing of the prior art of record relied upon in the rejection of claims under appeal.

U.S. Pat. 5,630,136 issued to Davidson et al., Filed Jun. 9,1995.

U.S. Pat. 5,644,768 issued to Periwal et al., Filed Dec. 9,1994.

(10) New Prior Art

No new prior art has been applied in this examiner's answer.

(11) Grounds of rejection.

The following grounds of rejection are applicable to the appealed claims:

The text of those sections of Title 35, US Code not included in this office action can be found in a prior Office Action.

In the final office action issued on 12/19/2000;

(I) The following is a quotation of 35 U.S.C. § 103 which forms the basis for all obviousness rejections set forth in this Office action:

Claims 1-7, are rejected under 35 U.S.C. § 103 as being unpatentable over Davidson et al. (U.S.Pat No. 5,630,136), in view of Periwal et al. (U.S.Pat No. 5,644,768).

As per claim 1, Davidson et al. teach:

- **a multi-tasking operating system** (**multi tasking operating system**, col. 4, lines 66-68, continued on col. 5, lines 1-11) for **managing simultaneous access** (**to synchronize multi thread processes within a multitasking computer system so that the thread can safely access a multi threading unsafe resource**, col. 2, lines 58-61), **serially reusable resource** (by using the baton object, **access to multi threading unsafe resources are efficiently serialized** and program deadlocks are minimized, col. 2, lines 49-54);

- at least one **resource** (**resource**, col. 2, lines 49-54);

- **plurality of threads** requesting access to said resource (threads that seeks to access the ,multi threading unsafe resource, col.2, lines 68 continued on col. 3, lines 1-14);

The reference of Davidson fails to explicitly teach of a stationary queue for allocating access to the resources one-by-one in order of request. Periwal in the same field of endeavor teaches of system resources to be shared among several clients concurrently and further teach of a **mutex record** that has three fields, i.e. (a) true mutex (handle or pointer to mutex), (b) mutex ID, and (c)mutex record count (col. 3, lines 31-44, and col. 8, lines 61-68, continued on col. 9, lines 1-10). Periwal, later teach of One-by-one in order of request (**later calls to acquire a mutex for a resource are processed by first checking the ID of the thread or process (currently requesting the mutex) against the mutex ID stored in the mutex record**. In the event that the two IDs match, the system simply increments the mutex reference count and permits the thread or process to continue execution. col. 3, lines 45-57).. **For the reason that** the thread or process does not deadlock on itself (See Periwal, background of invention, col. 2, lines 52-68, continued on col. 3, lines 1-6, and the teaching of Mutex and Semaphore for controlling access to shared **data structure among concurrent process or threads**).

Per claim 2, "the sleep code routine" is generally taught by Periwal in his background of the invention col. 2, lines 25-39 ("wait state" or "suspended state"). He also teaches of **mutex record ID** for sleeping process in col. 11, lines 45-68. Periwal teaches of awakening (re-animated in the claim language) the awakened process which carries out its processing task with the mutex in col. 11, lines 55-68.

Per claim 3, the reference of Periwal teaches of a reference count and **number of request for acquisition** and **number of**

releases in col. 3, lines 58-68, continued on col. 4, lines 1-5. By having the two, he teaches of the number of thread which have been denied access (as long as the number of requests for acquisition exceeds the number of releases, See Periwal, col. 3, lines 58-68). The reference of Periwal while teaches of number of release for acquisition and number or release, also teaches of the sleep code and wake-up code which are both responsive to wait counter and satisfied counter (number of request for acquisition exceeding the number of releases for that mutex).

Per claim 4, responsive to a request from a thread for a resource which is not available, creating a block identifier (thread ID for the mutex does not match that of the current process, the process is simply put into wait or sleep, col. 11, lines 25-53). Enabling subsequent wake up of the thread in fair order (for a process which has been put into this wait state, it will be awakened when the mutex is freed, col. 11, lines 25-33, and later, also back to col. 3, lines 58-68, the thread or process will retain acquisition as long as the number of requests for acquisition exceeds the number of releases for that mutex).

Per claim 5, the reference of Periwal teaches of a reference count and **number of request for acquisition** and **number of releases** in col. 3, lines 58-68, continued on col. 4, lines 1-5.

By having the two, he teaches of the number of thread which have been forced to wait and have been subsequently satisfied (as long as the number of **requests for acquisition exceeds** the **number of releases**, See Periwal, col. 3, lines 58-68). The reference of Periwal while teaches of number of request for acquisition and number or releases, also teaches of the sleep code and wake-up code which are both responsive to wait counter and satisfied counter (number of request for acquisition exceeding the number of releases for that mutex).

For the rejection of claim 6-7, please refer to the rejection of claims 4, and 5 combined.

(12) New Ground of Rejection

Examiner's Answer does not contain any new ground of rejection.

(13) Response to argument

(**I**) Applicant in his argument on page 8 argue that:

"Applicant... argue `On the other hand, applicant has provided a stationary queue. Such a queue, which includes two counters, insures not only fairness, but also on thread at a time is awakened and given access, so system performance in resource constrained scenarios is maintained. There is no thrashing of a run/wait queue in the kernel". In response it is submitted that, first fairness is interpreted as serializing threads, the reference of Davidson teaches of serializing threads in col. 2, lines 49-54... Regarding one thread at a time, Davidson teach `However, at most one of the baton objects for a given multi threading unsafe resource can own the baton at a given time."

"...The data structure which causes the thread access to be serialized is taught by Davidson, and to ensure the serializing of thread access, Periwal teaches of mutex record." (Office Action, 25 April 2001, page 4. Emphasis added.)

Applicant traverses this characterization of the teachings of Davidson and Periwal."

Examiner respectfully disagree. First, it should be noted that there is no recitation of "stationary queue" in claims 4-7. Therefore, this argument does not apply to claims 4-7. Regarding the recitation of "stationary queue" in claims 1-3, and the argument that the stationary queue has two counters. There is no recitation of any counters in claims 1. In claim 2, which is independent, again there is no counter recited. Claim 2, recite "sleep code" and "wake up code", where both of these limitations are taught by Periwal (See, rejection of claim 2, section 11, rejection of claims). Claim 3, recite "stationary queue", however, limitations should not be read into a claim. E.g., In re Prater, 415 F.2d 1393, 1404-5, 162 USPQ 541, 550-51 (CCPA 1969). Accord In re Zletz, 893 F.2d 319, 321, 13 USPQ2d 1320, 1322 (Fed. Cir. 1989) ("pending claims must be interpreted as broadly as their terms reasonably allow"). For an ordinary skill in the art a "stationary queue" is a queue, wherein, when data is entered it does not pop out conditionally. Appelant's attention is directed to page 9 of his specification and Table 1, and 2. In there, as it is indicated, a block ID is created using "number-forced-to-wait". This is not "stationary queue". For a person ordinary skill in the art, a stationary queue is one where data are in and out serially. **In re Prater, 162 USPQ 541, 550 (CCPA**

1969) "Reading a claim in the light of the specification" to thereby interpret limitations explicitly recited in the claim, is different from "reading limitations of the specification into a claim," to thereby narrow the scope of the claim by implicitly adding disclosed limitations in the claim. There is no elements of this special queue "stationary queue" in the claim. Additionally, claims in a pending application should be given their broadest reasonable interpretation. **In re Pearson,** 181 USPQ 641 (CCPA 1974). Claimed subject matter not the specification, is the measure of the invention. Disclosure contained in the specification can not be read into the claims for the purpose of avoiding prior art. **In re Sporck,** 55 CCPA 743, 386 F.2d924, 155 USPQ 687 (1986).

Regarding the word "fairness", the word is broad with no specific and it is subject to different interpretation by a person ordinary skill in the art. If the threads are awakend based on their priority, "fair" is priority awaikend thread. When the threads are awakend based on real-time calls, "fair" is "real-time" call awakening. Therefore, the order of fairness, depends on the way threads are awakened in different situation.

Regarding FIFO order in claim 2, Examiner belive that Appelant's characterization of FIFO is wrong. If page 9 of the specification is the true characterization of the invention, there is no FIFO order in this invention. As it is taught in tables 1, and 2 of the spec. (Page 9), when a wake-up call comes (line 10), the execution resumes. Also in tabe 2, "waiting in the stationary queue the longest" (line 24), "made ready to run by the operating system". This is not a FIFO at all, where FIFO, for an ordinary skill in the art means **First-in First-Out**. Which means thread **A** which is suspended and put into a queue must be out before thred **B** which is put in the queue after **A**. Therefore, the word FIFO is missunderstood by Appellant or at least it is not the term which is used by a person ordinary skill in the art.

(**II**) On page 9, appellant argue "Serial, in the sense of mutually exclusive, is what Davidson is teaching, but is not what applicant is claiming. Applicant's claims are drawn to "in order of request" (claim 1), "FIFO order" (claims 2 and 3), "fair order" (claims 4-7, or "next... in line" (claim 8).A mutex record does provide mutual exclusion among threads, but does not provide "fair" or "FIFO" order.". Later on page 11, lines 1-5, Appellant argue "Periwal teaches a mutex and a nesting mechanism for preventing deadlocks when accessing the mutex. This is done to prevent deadlock conditions but as described hereafter, as with Davidson, there is no teaching of the fair, or FIFO, ordering of threads.".

In response, it is submitted that "fair" for an ordinary skill in the art does not mean "FIFO". As it was discussed in section (I) above, the way, the threads are awakend in the specification is not a "FIFO".

(**III**) Appellant, on page 11, argue "Davidson and Periwal both fail to teach anything about enforcing order in the system. Periwal's reference to awakening "the next sleeping thread" (Col. 11, line 57) and to "subsequent threads or processes" (Col. 11, line 67) says nothing about enforcing order. No mention is made about how "next" or "subsequent" is determined, and in fact in Periwal (as well as in Davidson) that order is undefined, as is apparent from an examination of the code in the Source Code Appendix of Periwal.". Later on page 12, "In the case of a mutex, as taught in the on-line UNIX manual page at the URL noted above, "By default, if multiple threads are waiting for a mutex, the order of acquisition is undefined." (Emphasis added). By using a mutex structure in six of his seven operating system examples without any additional teaching **with respect to FIFO, fair,** etc. order, Periwal inherently teaches that the order of acquisition is undefined. Consequently, Periwal's system, as also Davidson's, necessarily involves the thrashing that applicant's invention avoids with a thread specific (**single, next in fair or FIFO order**) awakening.".

In response, Examiner disagree. Periwal, teaches of enforcing order in col. 3, lines 57-68, continued on col. 4, lines 1-5 (the thread or process will retain acquisition of the mutex as long as the number of requests for acquisition exceeds the number of release for that mutex. This create an order of awaikening of the subsequent thread and it is not left for the operating system to order the next thread to be awaikened.

(**IV**) On page 13 Appelant argue "In the case of a critical section, the seventh (in the above list) of the seven examples in the source code appendix of Periwal, as taught in the Microsoft Visual Studio Version 6.0 Help function on "EnterCriticalSection" (copied into a note dated 06/20/2001 09:40 AM from Bill Wilhelm to Shelley Beckstrand, copy attached) for mutual exclusion processing, each thread desiring to enter a critical section must first request and obtain ownership. When a thread having ownership, releases it, it is available for another thread. However, there is no structure or method taught by Periwal or provided by the Operating System for assuring that the "next" or "subsequent" thread to obtain ownership of the critical section is the next in fair, or FIFO, order. It is merely the "next" of all competing threads to win the race for ownership of the critical section. Applicants avoid such a race, or thrashing, by awakening only one thread, the one thread which applicant claims is in "FIFO order", "fair order", "in order of request", or "next thread in line".".

Examiner respectfully disagree. The one thread wich Applicant claims is in "FIFO", is not in "FIFO" order in the specification, as it was stated in section 11(**I**). Regarding fair order, as it was stated in section 11(**II**), it is interpreted just an order by a person orduinary skill in the art. Examiner believe that the ordering in this invention is not even "in order of request", because, assuming (arguendo), threads are set in the queue serially. Again assuming, the condition for wake-up call is not seriallized (say the number forced to wait is not equal to number satisfied), then the process that has been waiting in the stationary queue the longest is made ready to run by operating system" (specification, page 9). Does that means they are awakened according "in order of rewquest"?. Examiner does not think so. Because, the number forced to wait is not equal to the number satisfied. Additionally, they are forced to wait and not waited serially.

(**V**) Onpage 13-15, Appellant argue "The Examiner asserts that "there is no teaching of any (FIFO) queue or queue-like behavior without the use of·a real queue in the claims or even in the specification."

Applicant traverses. The stationary queue is described in the specification (See Section V, Summary of the Invention, supra), is explicitly claimed in claims 1-3, and elements of such a queue are present in the other claims, 48. This stationary queue, and the claimed elements of it, is provided to avoid the thrashing that is inherent in the MUTER process and assure FIFO order, fair order, order of request, or next in line order.

A stationary queue is not a queue in the normal use of the term "queue", but rather a structure which presents queue-like behavior without being a queue. Appellant's invention achieves queue-like behavior (first-come, first served) without the use of a real queue. As appellant explains in his specification:

In accordance with this invention, the solution to this scarce resource management problem produces the same result as a queue or FIFO, but there is no memory actually associated with the queue or FIFO structure. No data or identifiers are moved into or out of this queue, hence the term 'stationary'. Only the read and write pointers of the queue need be maintained since the data that would ordinarily be stored in the queue is impliedly the counter, or pointer, values themselves. (Page 5, lines 2-10.)

In response, regarding the argument of "queue like behavior without the use of a real queue (**First-come, first served**)", it is submitted that queue is missinterpreted as FIFO. Queue by definition is multielement data structure from which (by restrict definition) elements can be removed only in the same order in whcih they were inserted; i.e., it followes a first-in-fisrt-out (FIFO) constraint. However, there are also other types of queue in which removal is based on factors other than order of insertion. For example, in a priority queue the elements are removed according to some priority value assigned to each. As it was stated before, the order the threads are awaikened is not FIFO in the specification. Regarding the use of memory, examiner disagree. There is no operation in the computer without the use of a memory (either it is reading from memory or writing into the memory). The order of request is kept somewhere in the computer, otherwise, the computer cannot keep the orders magically. As it is specified in claim 8, the counters are keeping track of the pointers and the order of awaikening.

(14) Conclusion

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

Maid A. Banankhah

[illegible]
MAJID A. BANANKHAH
PRIMARY EXAMINER

Copies of the Examiner Answere, attached to the "order Returning Undocketed Appeal", dated Oct. 25, 02 have been Signed by the Conferees and show Appeal Conference have been Conducted.

M.B. 3/14 02